Sinhgad College of Engineering

Department Of Computer Engineering

Name Of Student:

Roll No.: Exam No.:

Class: Div: Batch:

Name Of Laboratory:

Soft Computing and Optimization Algorithm

Sr. No.	Name of Experiment	Start Date	Completion Date
1	Implementation of Fuzzy Operations: Union, Intersection, Complement and Difference operations on fuzzy sets.		
2	Implementation of Fuzzy Relation: by Cartesian product of any two fuzzy sets and perform max-min composition on fuzzy relations.		
3	Perform Functions of Logic Gates using McCulloh-Pitts Neural Network (model)		
4	Study and Implement optimization problem using Partial Swarm Optimization		
5	Mini Project -Implementation of Simple Genetic Application		

Cloud Computing

Sr. No.	Name of Experiment	Start Date	Completion Date
1	Installation and configuration of own Cloud.		
2	Implementation of Virtualization in Cloud Computing to Learn Virtualization Basics, Benefits of Virtualization in Cloud using Open Source Operating System		
3	Case study on Amazon EC2 to learn about Amazon EC2, Amazon Elastic Compute Cloud is a central part of Amazon.com's cloud computing platform, Amazon Web Services. How EC2 allows users torrent virtual computers on which to run their own computer applications.		
4	Case study on Microsoft azure to learn about Microsoft Azure is a cloud computing platform and infrastructure, created by Microsoft, for building, deploying and managing applications and services through a global network of Microsoft-managed data centre's. How it works, different services provided by it.		
5	Write a Program to Create, Manage and groups User accounts in own Cloud by Installing Administrative Features.		
6	Assignment to install and configure Google App Engine.		
7	Mini Project - Real time implementation for data streaming using google cloud pubsub and bigquery		

Assignment No.1

Code of Program(write our name in comments) –

```
#fuzzy sets operations'''
Implement Union, Intersection, Complement and Difference operations on fuzzy sets.
Also create fuzzy relation by Cartesian product of any two fuzzy sets
and perform max-min composition on any two fuzzy relations. '''

#Here we define two sets young(x) and middleAged(x)def student(x):
        if x<=20:
                return 1 elif
        20<x<=30:
                return (30-x)/(30-20)elif x>30:
                return 0


def doJob(x):
        if x>30:
                return 1elif
        x<=20:
                return 0 elif
        20<x<=30:
                return (x-20)/(10)


class Fuzzy:
        def__init__(self,item,membership):self.val =    item
                self.membership = membership

        def return_membership(self):
              return self.membershipimport

random

x1 = random.sample(range(15, 40), 20)x2 = x1

set1 = [] #studentset2 = []
#doJob

print(x1)
print(x2)

[28, 35, 25, 29, 34, 30, 16, 32, 19, 39, 38, 33, 37, 23, 21, 27, 18, 17, 22, 20]
[28, 35, 25, 29, 34, 30, 16, 32, 19, 39, 38, 33, 37, 23, 21, 27, 18, 17, 22, 20]
for member in x1:
        val = student(member) obj =
        Fuzzy(member,val)
        set1.append(obj)

for member in x2:
        val   =   doJob(member) obj =
        Fuzzy(member,val)
        set2.append(obj)
```

```
 for obj in set1: print(obj.val,':',obj.membership)

print('================================')

 for obj in set2: print(obj.val,':',obj.membership)
```

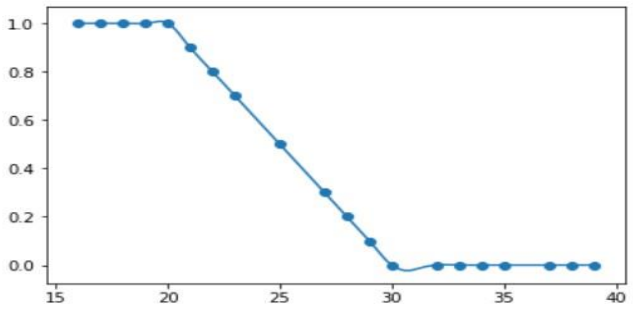
```
28 : 0.2
35 : 0
25 : 0.5
29 : 0.1
34 : 0
30 : 0.0
16 : 1
32 : 0
19 : 1
39 : 0
38 : 0
33 : 0
37 : 0
23 : 0.7
21 : 0.9
27 : 0.3
18 : 1
17 : 1
22 : 0.8
20 : 1
================================28 :
0.8
35 : 1

25 : 0.5
29 : 0.9
34 : 1
30 : 1.0
16 : 0
32 : 1
19 : 0
39 : 1
38 : 1
33 : 1
37 : 1
23 : 0.3
21 : 0.1
27 : 0.7
18 : 0
17 : 0
22 : 0.2
20 : 0
```

```
student_x = [] student_y
= [] for obj in set1:
        student_x.append(obj.val) student_y.append(obj.membership)
job_x=[]
job_y=[]
 for obj in set2: job_x.append(obj.val)
        job_y.append(obj.membership)

print(student_x,student_y,job_x,job_y)
```

[28, 35, 25, 29, 34, 30, 16, 32, 19, 39, 38, 33, 37, 23, 21, 27, 18, 17, 22, 20]
[0.2, 0, 0.5, 0.1, 0, 0.0, 1, 0, 1, 0, 0, 0, 0, 0.7, 0.9, 0.3, 1, 1, 0.8, 1] [28,
35, 25, 29, 34, 30, 16, 32, 19, 39, 38, 33, 37, 23, 21, 27, 18, 17, 22, 20] [0.8,
1, 0.5, 0.9, 1, 1.0, 0, 1, 0, 1, 1, 1, 1, 0.3, 0.1, 0.7, 0, 0, 0.2, 0]

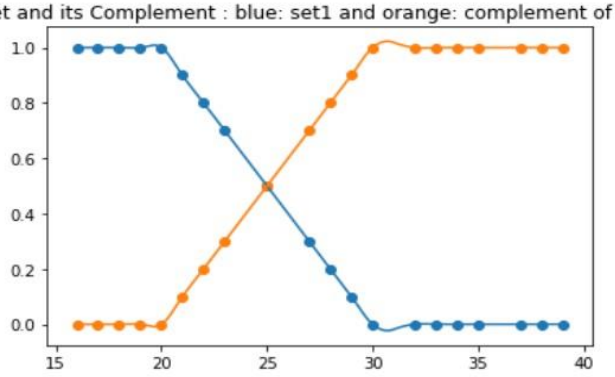
```
import numpy as np
import matplotlib.pyplot as plt

from scipy.interpolate import interp1d

x=np.array(student_x)
y=np.array(student_y)

x_new = np.linspace(x.min(), x.max(),500)

f = interp1d(x, y, kind='quadratic')y_smooth=f(x_new)

#=====================================

x1=np.array(job_x)
y1=np.array(job_y)

x_new1 = np.linspace(x1.min(), x1.max(),500)
f1 = interp1d(x1, y1, kind='quadratic')
y_smooth1=f1(x_new1)

plt.plot (x_new,y_smooth)plt.scatter
(x, y)
plt.plot (x_new1,y_smooth1)plt.scatter
(x1, y1)

#complement#on
set 1
import numpy as np
import matplotlib.pyplot as plt

from scipy.interpolate import interp1d

x=np.array(student_x)
y=np.array(student_y)

x_new = np.linspace(x.min(), x.max(),500)

f = interp1d(x, y, kind='quadratic')y_smooth=f(x_new)

plt.plot (x_new,y_smooth)plt.scatter
(x, y)
```

<matplotlib.collections.PathCollection at 0x26f9fa972e8>



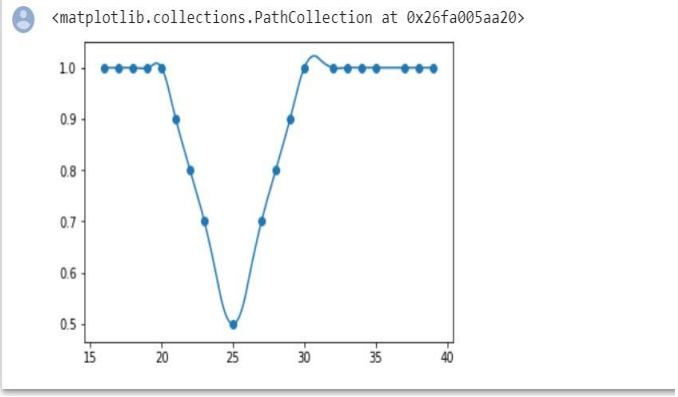

```
NOTset1   =   [] for obj in
set1:
        new_obj = Fuzzy(obj.val,1- obj.membership)
        NOTset1.append(new_obj)

student_x1 = [] student_y1   =
[] for obj in NOTset1:
        student_x1.append(obj.val) student_y1.append(obj.membership)

x_compli=np.array(student_x1)y_compli=np.array(student_y1)

x_new_compli = np.linspace(x_compli.min(), x_compli.max(),500)f2 = interp1d(x_compli,

y_compli, kind='quadratic')
x=np.array(student_x)
y=np.array(student_y)

x_new = np.linspace(x.min(), x.max(),500)

f = interp1d(x, y, kind='quadratic')y_smooth=f(x_new)

plt.plot (x_new,y_smooth)plt.scatter (x,
y)

x_compli=np.array(student_x1)y_compli=np.array(student_y1)

x_new_compli = np.linspace(x_compli.min(), x_compli.max(),500)

f2 = interp1d(x_compli, y_compli, kind='quadratic')
y_smooth_compli=f2(x_new_compli)
plt.title('Set and its Complement : blue: set1 and orange: complement of set1')plt.plot (x_new_compli,y_smooth_compli)
plt.scatter (x_compli, y_compli)
```

<matplotlib.collections.PathCollection at 0x26f9fffca20>



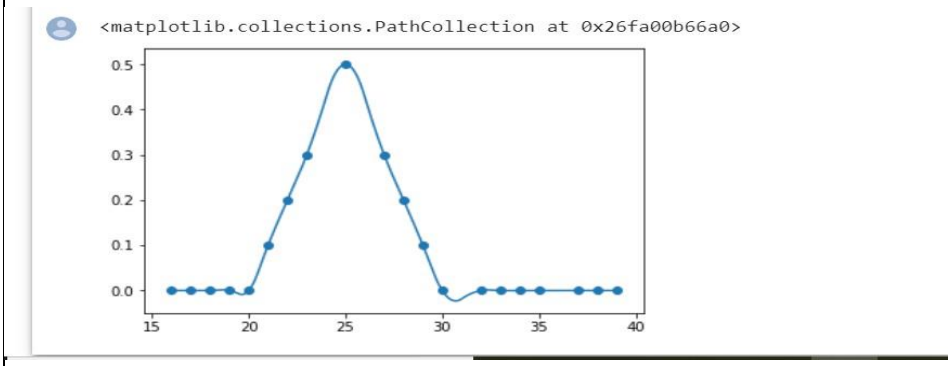

```
#union import
mathunion = []
for obj1,obj2 in zip(set1,set2):
        new_obj = Fuzzy(obj1.val,max(obj1.membership,obj2.membership))union.append(new_obj)

Ux = []
Uy = []
  for obj in union: Ux.append(obj.val)
        Uy.append(obj.membership)

x=np.array(Ux)
y=np.array(Uy)

x_new = np.linspace(x.min(), x.max(),500)

f = interp1d(x, y, kind='quadratic')y_smooth=f(x_new)


plt.plot (x_new,y_smooth)plt.scatter
(x, y)
```

<matplotlib.collections.PathCollection at 0x26fa005aa20>




```
#intersection intersection
= []
for obj1,obj2 in zip(set1,set2):
        new_obj = Fuzzy(obj1.val,min(obj1.membership,obj2.membership))
        intersection.append(new_obj)
```

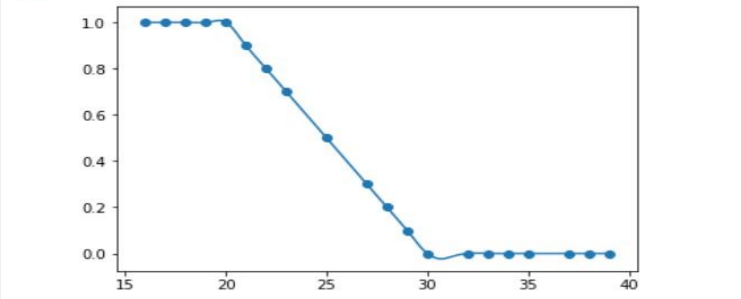
```
Ix = []
Iy = []
  for obj in intersection: Ix.append(obj.val)
          Iy.append(obj.membership)



x=np.array(Ix)


y=np.array(Iy)

x_new = np.linspace(x.min(), x.max(),500)

f = interp1d(x, y, kind='quadratic')y_smooth=f(x_new)


plt.plot (x_new,y_smooth)plt.scatter
(x, y)
```




```
#difference
#difference of 2 sets F1 and F2 is# F1 - F2 = F1
intersect |-F2|


# set2 complement
NOTset2 = []
for obj in set2:
          new_obj = Fuzzy(obj.val,1- obj.membership)
          NOTset2.append(new_obj)


intersection1 = []
for obj1,obj2 in zip(set1,NOTset2):
          new_obj = Fuzzy(obj1.val,min(obj1.membership,obj2.membership))
          intersection1.append(new_obj)

Ix1 = []
Iy1 = []
  for obj in intersection1: Ix1.append(obj.val)
          Iy1.append(obj.membership)
```

```
x=np.array(Ix1)
y=np.array(Iy1)

x_new = np.linspace(x.min(), x.max(),500)

f = interp1d(x, y, kind='quadratic')y_smooth=f(x_new)

plt.plot (x_new,y_smooth)plt.scatter
(x, y)
```

```
<matplotlib.collections.PathCollection at 0x26fa10e6860>
```



Assignment No.2

Code of Program –

```
# MIN-MAX composition of 2 fuzzy relatons'''
Let 2 relations be R and S and given as follows'''

  R = [ [0.62,0.45,0.4],
        [0.12,0.1,0,34],
        [0.43,0.27,0.9]
]


  S = [ [0.11,0.33,0.98],
        [0.23,0.37,0.74],
        [0.6,0.5,0.19]
]

#min-max composition will be N*N matrix where N is dimension of both R and SN = 3
min_max = []

for i in range(N):
        List = []
        for j in range(N):
                #ith row and jth columnI = R[i]
                new = []
                for k in range(N): new.append(min(I[k],S[k][j]))
                List.append(max(new))
        min_max.append(List)



print('R : ')
for i in range(N):
        for j in range(N): print(R[i][j],' ',end='')
       print() print('================================')

print('S:')
for i in range(N):
        for j in range(N): print(S[i][j],' ',end=' ')
        print()
print('=========MIN-MAX COMPOSITION IS: ==========')
for i in range(N):
        for j in range(N): print(min_max[i][j],' ',end='')
        print()
```

R :
0.62 0.45 0.4
0.12 0.1 0
0.43 0.27 0.9
==========S:
0.11 0.33 0.98
0.23 0.37 0.74
0.6 0.5 0.19
==========MIN-MAX COMPOSITION IS: ==========0.4
0.4 0.62
0.11 0.12 0.12
0.6 0.5 0.43

Assignment No.3

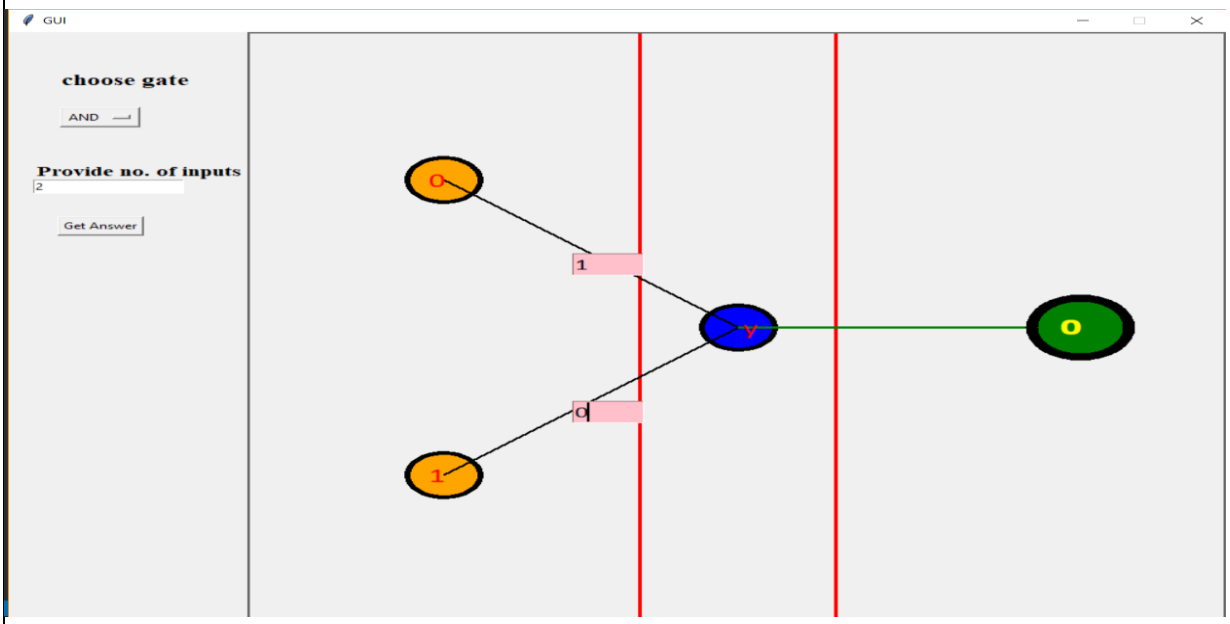
Code of Program –

```
import tkinter
from tkinter import *inputs = []

number_of_gates = 0 outx=0

outy=0 label_id = -1

#create application windowwindow =
Tk()

#to rename the title of the window by default it is 'tk'window.title('GUI')

#set window size
window.geometry('%dx%d+%d+%d' % (1000, 800, 0, 0))

#window can't be resized window.resizable(False,False)

 def draw_diagram():
        c.delete("all")

        number_of_gates = int(e.get())

        #print(number_of_gates) #print(type_of_gate)
        canvas_width = c.winfo_width() canvas_height
        = c.winfo_height()

       #we need to divide entire canvas into 3 columns in order to display inputnodes 'y' node and output line
        #all of them are not of same width#input : 2/5
        'y':1/5 line:2/5 import math
        #separate 3 parts using line
        x = math.ceil(canvas_width * 2 / 5)
        c.create_line(x,0,x,canvas_height,fill="red",width=3)x = x +
        math.ceil(canvas_width * 1 / 5)
        c.create_line(x,0,x,canvas_height,fill="red",width=3)

        #draw circles for gate
        part_per_circle = canvas_height // number_of_gatesx =
        math.ceil(canvas_width * 1 / 5)
        y = part_per_circle // 2r = 30
        for i in range(part_per_circle):
                c.create_oval(x-r, y-r, x+r, y+r,fill="orange",width=5)
                y += part_per_circle
```

```
        #draw 'y'
        x1 = math.ceil(canvas_width * 2 / 5) + math.ceil(canvas_width * 1 / 10)y1 = canvas_height // 2
        r = 30
        c.create_oval(x1-r, y1-r, x1+r, y1+r,fill="blue",width=5)#draw input lines

        x = math.ceil(canvas_width * 1 / 5)y =
        part_per_circle // 2
        for i in range(number_of_gates): c.create_line(x,y,x1,y1,fill="black",width='2')
                c.create_text(x-5,y,text=i,font=("Calibri",20),fill="red") e1 =
                tkinter.Entry(c,width=5,bg="pink",font=("Calibri",16))t1 = (x1+x-30)//2
                t2 = (y1+y)//2 e1.place(x = t1 ,y =
                t2)inputs.append(e1)

                c.update()
                y += part_per_circle

        #draw output line
        x = math.ceil(canvas_width * 3 / 5) + math.ceil(canvas_width * 1 / 5)y = canvas_height // 2
        c.create_line(x1,y1,x,y,fill="green",width='3')
        c.create_text(x1+10,y1,text='y',font=("Calibri",20),fill="red") c.update()

        #draw final answer circlex = x + 40

        c.create_oval(x-40,y-40,x+40,y+40,fill="green",width=10)



        submit.place_forget()
        getAns.place(x=40,y=250) #end of
        function definition#


def Mc_culloch_pits():
        '"WEIGHT OF EVERY INPUT EDGE IS +1 HENCE OUTPUT DEPENDS UPON INPUT GIVENBY USER
ONLY"'
        number_of_gates = int(e.get()) print('Number of gates =
        ',number_of_gates)type_of_gate = var.get()
        #calculate input weighted sumsum = 0
        for item in inputs:
                sum+= int(item.get())
        print('sum = ',sum) output = 'NA'
        if type_of_gate == "AND":
                #threshold definition
                threshold = number_of_gates * 1if sum >=
                threshold:
                        output = 1else:
```

```
                output = 0

        elif type_of_gate == "OR":#threshold
                definition threshold = 1 * 1
                if sum >= threshold:output =
                        1
                else:
                        output = 0

        elif type_of_gate == "NAND":#threshold
                definition
                threshold = number_of_gates * 1if sum >=
                threshold:
                        output = 0else:
                        output = 1

        elif type_of_gate == "NOR":#threshold
                definition threshold = 1 * 1
                if sum >= threshold:output =
                        0
                else:
                        output = 1

        import math
        outx = math.ceil(c.winfo_width() * 3 / 5) + math.ceil( c.winfo_width() * 1
/ 5) + 40
        outy = c.winfo_height() // 2
        print('For type of gate = ',type_of_gate,' answer is = ',output)ans.place(x=outx-20,y=outy-20)
        v1.set(output) #c.create_text(outx,outy,text=output,font=("Calibri",30),fill="yellow")c.update()


c = tkinter.Canvas(window, height=800, width=800)c.pack(side="right")

#text label for GATES
gates = tkinter.Label(window,text="choose gate") gates.configure(font=("Times
New Roman", 16, "bold"))gates.place(x=40,y=50)


#option menu for names of GATEsvar =
StringVar(window) var.set("AND") # initial
value
option = OptionMenu(window, var, "AND", "OR", "NAND", "NOR")option.place(x=40,y=100)

#text label for input of number of gates
number = tkinter.Label(window,text="Provide no. of inputs")number.configure(font=("Times New Roman", 14,
"bold")) number.place(x=20,y=175)

#number of inputs
e = tkinter.Entry(window)
```

```
e.place(x=20,y=200)

#submit button
submit = tkinter.Button(window,text = "SUBMIT",command=draw_diagram)submit.place(x=40,y=250)

#getAnswer button
getAns = tkinter.Button(window,text = "Get Answer",command=Mc_culloch_pits)getAns.place_forget()

#answer
v1 = StringVar()
ans = tkinter.Label(c,textvariable=v1,bg="green",font = "Verdana 20bold",fg="yellow")
ans.place_forget()


window.mainloop()
```



Assignment No.4

Code of Program –

```
# Perceptron Algorithm on the Sonar Datasetfrom random import
seed
from random import randrangefrom csv
import reader

# Load a CSV file
 def load_csv(filename):dataset =
     list()
     with open(filename, 'r') as file:csv_reader =
         reader(file)
         for row in csv_reader:if not row:
                 continue dataset.append(row)
     return dataset

# Convert string column to float
 def str_column_to_float(dataset, column):for row in dataset:
         row[column] = float(row[column].strip())

# Convert string column to integer
 def str_column_to_int(dataset, column): class_values = [row[column] for
     row in dataset]unique = set(class_values)
     lookup = dict()
     for i, value in enumerate(unique):lookup[value] = i
     for row in dataset:
         row[column] = lookup[row[column]]return
     lookup

# Split a dataset into k folds
 def cross_validation_split(dataset, n_folds):dataset_split = list()
     dataset_copy = list(dataset)
     fold_size = int(len(dataset) / n_folds)for i in
     range(n_folds):
         fold = list()
         while len(fold) < fold_size:
             index = randrange(len(dataset_copy))
             fold.append(dataset_copy.pop(index))
         dataset_split.append(fold)return
     dataset_split

# Calculate accuracy percentage
 def accuracy_metric(actual, predicted):correct = 0
     for i in range(len(actual)):
         if actual[i] == predicted[i]:correct += 1
     return correct / float(len(actual)) * 100.0
```

```
# Evaluate an algorithm using a cross validation split
 def evaluate_algorithm(dataset, algorithm, n_folds, *args):folds =
     cross_validation_split(dataset, n_folds)
     scores = list() for fold in
     folds:
         train_set = list(folds)
         train_set.remove(fold)

         train_set = sum(train_set, [])test_set = list()
         for row in fold: row_copy =
             list(row)
             test_set.append(row_copy)
             row_copy[-1] = None
         predicted = algorithm(train_set, test_set, *args)actual = [row[-1] for row
         in fold]
         accuracy = accuracy_metric(actual, predicted)
         scores.append(accuracy)
     return scores

# Make a prediction with weightsdef
predict(row, weights):
     activation =    weights[0] for i in
     range(len(row)-1):
         activation += weights[i + 1] * row[i] return 1.0 if activation >=
     0.0 else 0.0


# Estimate Perceptron weights using stochastic gradient descentdef train_weights(train, l_rate,
n_epoch):
     weights = [0.0 for i in range(len(train[0]))]for epoch in
     range(n_epoch):
         for row in train:
             prediction = predict(row, weights)error = row[-1] -
             prediction
             weights[0] = weights[0] + l_rate * errorfor i in
             range(len(row)-1):
                 weights[i + 1] = weights[i + 1] + l_rate * error * row[i]return weights

# Perceptron Algorithm With Stochastic Gradient Descentdef perceptron(train, test,
l_rate, n_epoch):
     predictions = list()
     weights = train_weights(train, l_rate, n_epoch)for row in test:
         prediction = predict(row, weights)
         predictions.append(prediction)
     return(predictions)
```

```
# Test the Perceptron algorithm on the sonar datasetseed(1)
# load and prepare data
filename = 'D:\\BE_4434\\sonar-data-set\\sonar.all-data.csv'dataset = load_csv(filename)
 for i in range(len(dataset[0])-1):
     str_column_to_float(dataset, i)
# convert string class to integers str_column_to_int(dataset,
len(dataset[0])-1)# evaluate algorithm
n_folds = 3
l_rate = 0.01
n_epoch = 500



scores = evaluate_algorithm(dataset, perceptron, n_folds, l_rate, n_epoch)print('Scores: %s' % scores)
print('Mean Accuracy: %.3f%%' % (sum(scores)/float(len(scores))))
```

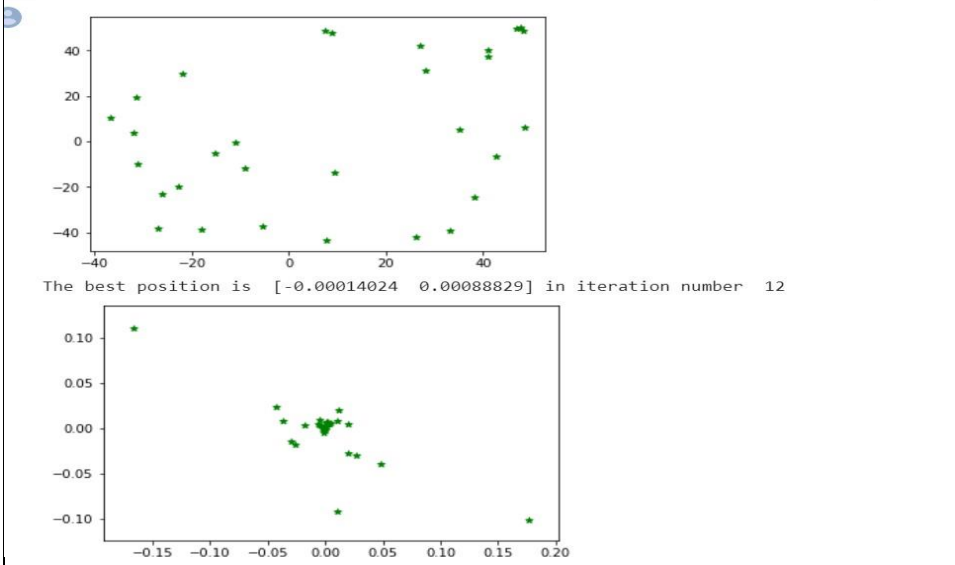
```
Scores: [81.15942028985508, 69.56521739130434, 62.31884057971014]
Mean Accuracy: 71.014%
```

Assignment No.5

Code of Program –

```
import random import numpy as np

W = 0.5
c1 = 2
c2 = 2
target = 1

n_iterations = 50 target_error = 1e-6n_particles = 30

 def visualize(particle_):x=[]
        y=[]
        for part in particle_:x.append(part[0])
                y.append(part[1])
        import matplotlib.pyplot as plt
        plt.scatter(x, y, label= "stars", color= "green", marker= "*", s=30)plt.show()

#function that models the problemdef
fitness_function(position):
        return position[0]**2 + position[1]**2 + 1

particle_position_vector = np.array([np.array([(-1) ** (bool(random.getrandbits(1))) * random.random()*50, (- 1)**(bool(random.getrandbits(1))) * random.random()*50]) for _ in range(n_particles)])
visualize(particle_position_vector) pbest_position = particle_position_vector
pbest_fitness_value = np.array([float('inf') for _ in range(n_particles)])gbest_fitness_value = float('inf')
gbest_position = np.array([float('inf'), float('inf')])

velocity_vector = ([np.array([0, 0]) for _ in range(n_particles)])iteration = 0
 while iteration < n_iterations: for i in
        range(n_particles):
                fitness_cadidate = fitness_function(particle_position_vector[i])#print(fitness_cadidate, ' ', particle_position_vector[i])

                if(pbest_fitness_value[i] > fitness_cadidate): pbest_fitness_value[i] = fitness_cadidate pbest_position[i] = particle_position_vector[i]
                if(gbest_fitness_value > fitness_cadidate): gbest_fitness_value = fitness_cadidate gbest_position = particle_position_vector[i]

        if(abs(gbest_fitness_value - target) < target_error):break
```

```
        for i in range(n_particles):
            new_velocity = (W*velocity_vector[i]) + (c1*random.random()) * (pbest_position[i] -
particle_position_vector[i]) + (c2*random.random()) * (gbest_position- particle_position_vector[i])
             new_position = new_velocity + particle_position_vector[i]particle_position_vector[i]
             = new_position

        iteration = iteration + 1

print("The best position is ", gbest_position, "in iteration number ", iteration)visualize(particle_position_vector)
```



Mini Project (SCOA)

Code of Program –

```
import random

def random_chromosome(size): #making random chromosomes
    return [ random.randint(1, nq) for _ in range(nq) ]

def fitness(chromosome):
    horizontal_collisions = sum([chromosome.count(queen)-1 for queen in chromosome])/2
    diagonal_collisions = 0

    n = len(chromosome)
    left_diagonal = [0] * 2*n
    right_diagonal = [0] * 2*n
    for i in range(n):
        left_diagonal[i + chromosome[i] - 1] += 1
        right_diagonal[len(chromosome) - i + chromosome[i] - 2] += 1

    diagonal_collisions = 0
    for i in range(2*n-1):
        counter = 0
        if left_diagonal[i] > 1:
            counter += left_diagonal[i]-1
        if right_diagonal[i] > 1:
            counter += right_diagonal[i]-1
        diagonal_collisions += counter / (n-abs(i-n+1))

    return int(maxFitness - (horizontal_collisions + diagonal_collisions)) #28-(2+3)=23

def probability(chromosome, fitness):
    return fitness(chromosome) / maxFitness

def random_pick(population, probabilities):
    populationWithProbabilty = zip(population, probabilities)
    total = sum(w for c, w in populationWithProbabilty)
    r = random.uniform(0, total)
    upto = 0
    for c, w in zip(population, probabilities):
        if upto + w >= r:
            return c
        upto += w
    assert False, "Shouldn't get here"

def reproduce(x, y): #doing cross_over between two chromosomes
    n = len(x)
    c = random.randint(0, n - 1)
    return x[0:c] + y[c:n]

def mutate(x):  #randomly changing the value of a random index of a chromosome
    n = len(x)
    c = random.randint(0, n - 1)
    m = random.randint(1, n)
    x[c] = m
```

```
    return x

def genetic_queen(population, fitness):
    mutation_probability = 0.03
    new_population = []
    probabilities = [probability(n, fitness) for n in population]
    for i in range(len(population)):
        x = random_pick(population, probabilities) #best chromosome 1
        y = random_pick(population, probabilities) #best chromosome 2
        child = reproduce(x, y) #creating two new chromosomes from the best 2 chromosomes
        if random.random() < mutation_probability:
            child = mutate(child)
        print_chromosome(child)
        new_population.append(child)
        if fitness(child) == maxFitness: break
    return new_population

def print_chromosome(chrom):
    print("Chromosome = {},  Fitness = {}"
        .format(str(chrom), fitness(chrom)))

if __name__ == "__main__":
    nq = int(input("Enter Number of Queens: ")) #say N = 8
    maxFitness = (nq*(nq-1))/2  # 8*7/2 = 28
    population = [random_chromosome(nq) for _ in range(100)]

    generation = 1

    while not maxFitness in [fitness(chrom) for chrom in population]:
        print("=== Generation {} ===".format(generation))
        population = genetic_queen(population, fitness)
        print("")
        print("Maximum Fitness = {}".format(max([fitness(n) for n in population])))
        generation += 1
    chrom_out = []
    print("Solved in Generation {}!".format(generation-1))
    for chrom in population:
        if fitness(chrom) == maxFitness:
            print("");
            print("One of the solutions: ")
            chrom_out = chrom
            print_chromosome(chrom)

    board = []

    for x in range(nq):
        board.append(["x"] * nq)

    for i in range(nq):
        board[nq-chrom_out[i]][i]="Q"



    def print_board(board):
        for row in board:
```

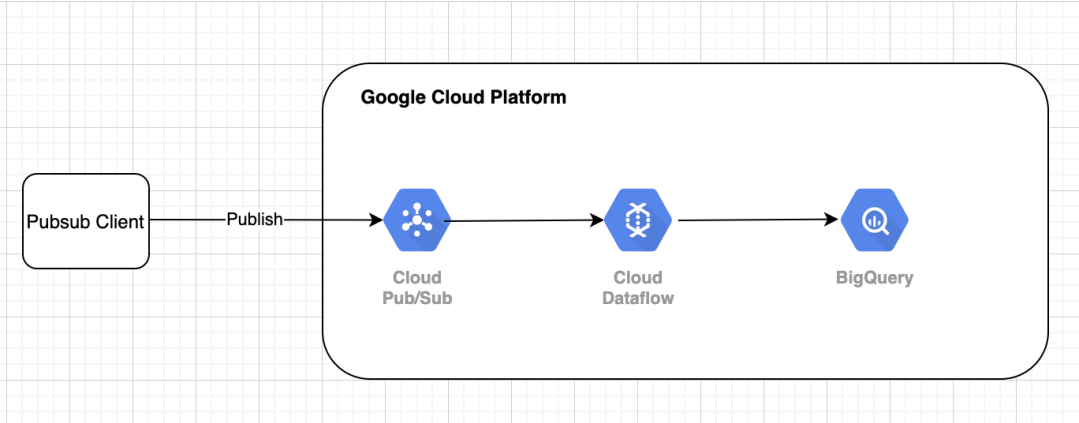
```
        print (" ".join(row))

    print()
    print_board(board)
```

Output :

```
Maximum Fitness = 14
=== Generation 3 ===
Chromosome = [5, 2, 2, 6, 3, 3],  Fitness = 13
Chromosome = [1, 6, 4, 3, 6, 4],  Fitness = 12
Chromosome = [1, 6, 6, 3, 6, 2],  Fitness = 11
Chromosome = [2, 4, 6, 4, 4, 5],  Fitness = 11
Chromosome = [1, 5, 2, 6, 3, 3],  Fitness = 14
Chromosome = [2, 2, 6, 6, 4, 6],  Fitness = 10
Chromosome = [2, 2, 4, 1, 2, 5],  Fitness = 11
Chromosome = [2, 4, 6, 1, 3, 1],  Fitness = 14
Chromosome = [5, 6, 2, 5, 6, 3],  Fitness = 12
Chromosome = [4, 6, 6, 3, 3, 2],  Fitness = 12
Chromosome = [4, 2, 5, 5, 2, 1],  Fitness = 12
Chromosome = [2, 6, 2, 3, 3, 3],  Fitness = 10
Chromosome = [6, 4, 2, 1, 2, 3],  Fitness = 12
Chromosome = [6, 4, 2, 5, 3, 5],  Fitness = 13
Chromosome = [5, 1, 4, 4, 5, 2],  Fitness = 12
Chromosome = [2, 4, 2, 1, 3, 5],  Fitness = 13
Chromosome = [2, 4, 6, 1, 3, 5],  Fitness = 15

Maximum Fitness = 15
Solved in Generation 3!

One of the solutions:
Chromosome = [2, 4, 6, 1, 3, 5],  Fitness = 15

x x Q x x x
x x x x x Q
x Q x x x x
x x x x Q x
Q x x x x x
x x x Q x x
```

Mini Project

Architecture :



Code :-

Food ordering csv

food_daily

Saved to Drive

File Edit View Insert Format Data Tools Extensions Help Last edit was seconds ago

Share

Customer_id

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Customer_id	date	time	order_id	items	amount	mode	restaurnt	Status	ratings	feedback		
2	OXJY167254JK	11-11-2020	8.11.21	854A854	Chow M?ein:	65	Cash	Sadabahar	Delivered	5	Awesome experience		
3	JXJY167254JK	11-11-2020	8.12.20	654S654	PiZza:Manch?ur	197	Wallet	Emperial	Delivered	2	Late delivery		
4	JXJY167254JK	11-11-2020	8.12.21	2444454	Noodles:Pizza:R	97	Card	Sadabahar	Delivered	3	Stale food		
5	XVTR474839TP	11-11-2020	8.12.22	397T397	Fried Rice:salaD	46	Card	Emperial	Not Delivered	3	Complicated procedure		
6	UFDF355524DM	11-11-2020	8.12.23	428K428	noo%dles:	71	Card	Delhi pot	Delivered	1	Food not good		
7	FRBT691245BA	11-11-2020	8.12.24	437M437	Laccha Prantha:	29	Online	Eat out	delivered	1	Stale food		
8	QTSI285174SG	11-11-2020	8.12.25	106S106	Fried Rice:	59	Cash	Toit	Delivered	2	Complicated procedure		
9	SAMV824387M\	11-11-2020	8.12.26	751C751	Quiche:	28	Wallet	Khana Khazana	Delivered	1	Difficult to order		
10	JXJY167254JK	11-11-2020	8.12.27	5589654	Quiche:Crispy O	87	Card	Johnahs	Delivered	2	Late delivery		
11	JXJY167254JK	11-11-2020	8.12.28	8548654	Noodles:Pizza:R	97	Card	Onesta	Cancelled	1	Stale food		
12	JXJY167254JK	11-11-2020	8.12.29	954S654	Fryams:Manchu	77	Wallet	Johnahs	Delivered	3	Late delivery		
13	JXJY167254JK	11-11-2020	8.12.30	154S644	Quiche:Noodles:	123	Card	Eat out	On Hold	4	Will order again		
14	LRKI603654KL	11-11-2020	8.12.31	742G742	Roti:	86	Card	Little Cafe	Delivered	1	Difficult to order		
15	FRBT691245BA	11-11-2020	8.12.32	103Y103	Manchurian:	94	Online	Plan B	delivered	1	Late delivery		
16	FSAM929386AC	11-11-2020	8.12.33	208U208	Paneer tikka:	33	Wallet	Emperial	Delivered	1	Difficult to order		
17	FRBT691245BA	11-11-2020	8.12.34	819Q819	Regular:	86	Cash	Khana Khazana	Cancelled	4	Cheap and best		
18	UFDF355524DM	11-11-2020	8.12.35	107K107	Biryani:	50	Card	Eat out	Delivered	4	Delicious food		
19	FSAM929386AC	11-11-2020	8.12.36	671T671	Idli:	82	Card	Delhi pot	On Hold	5	Good service		
20	QTSI285174SG	11-11-2020	8.12.37	195K195	Crispy Onion Rir	40	Cash	Delhi pot	Delivered	1	High price		
21	QTSI285174SG	11-11-2020	8.12.38	952U952	pickle:	29	Online	Toit	Delivered	3	Complicated procedure		
22	LRKI603654KL	11-11-2020	8.12.39	833F833	Chikko shake:	33	Cash	Emperial	Delivered	3	Coupon applied		

File_to_pubsub.py

```
from google.cloud import pubsub_v1
import time
```

```
publisher = pubsub_v1.PublisherClient()

topic_name = 'projects/<project ID>/topics/data_stream_from_file'
try:
    publisher.create_topic(topic_name)

except:
    print ('Topic already exists')

with open('food_daily.csv') as f_in:
    for line in f_in:
        # Data must be a bytestring
        data = line
        future = publisher.publish(topic_name, data=data)
        print(future.result())
        time.sleep(1)
```

Code_written_pubsub.py

```
#project-id:dataset_id.table_id
delivered_table_spec = 'bigquery-demo-
285417:dataset_food_orders.delivered_orders'
#project-id:dataset_id.table_id
other_table_spec = 'bigquery-demo-
285417:dataset_food_orders.other_status_orders'

import apache_beam as beam
from apache_beam.options.pipeline_options import PipelineOptions,
StandardOptions
import argparse
from google.cloud import bigquery


parser = argparse.ArgumentParser()

'''
format: projects/<project-id>/topics/<topic>
input_topic = 'projects/test-pipeline-253103/topics/test-pipeline-topic'
'''
parser.add_argument('--input',
                    dest='input',
                    required=True,
                    help='Input file to process.')

path_args, pipeline_args = parser.parse_known_args()

inputs_pattern = path_args.input

options = PipelineOptions(pipeline_args)
```

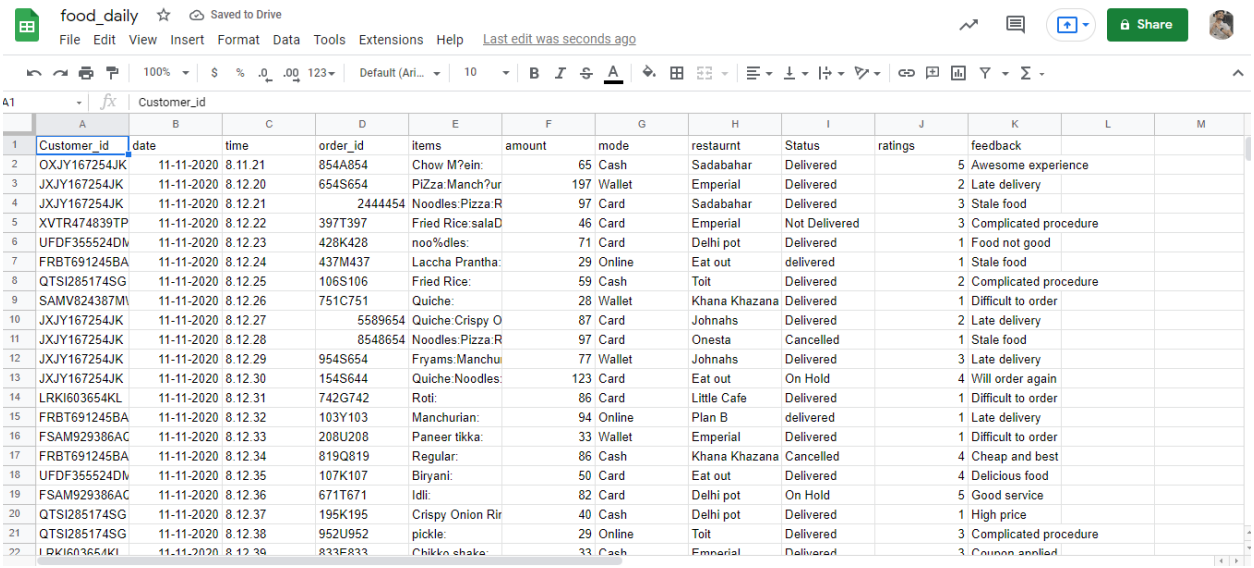
```
options.view_as(StandardOptions).streaming = True
#options.streaming = True # PubSub only works in streaming mode

p = beam.Pipeline(options = options)

def remove_last_colon(row):       # OXJY167254JK,11-09-
2020,8:11:21,854A854,Chow M?ein:,65,Cash,Sadabahar,Delivered,5,Awesome
experience
    cols = row.split(',')         # [(OXJY167254JK) (11-11-2020)
(8:11:21) (854A854) (Chow M?ein:) (65) (Cash) ....]
    item = str(cols[4])                # item = Chow M?ein:

    if item.endswith(':'):
        cols[4] = item[:-1]       # cols[4] = Chow M?ein

    return ','.join(cols)         # OXJY167254JK,11-11-
2020,8:11:21,854A854,Chow M?ein,65,Cash,Sadabahar,Delivered,5,Awesome
experience

def remove_special_characters(row):    # oxjy167254jk,11-11-
2020,8:11:21,854a854,chow m?ein,65,cash,sadabahar,delivered,5,awesome
experience
    import re
    cols = row.split(',')              # [(oxjy167254jk) (11-11-2020)
(8:11:21) (854a854) (chow m?ein) (65) (cash) ....]
    ret = ''
    for col in cols:
        clean_col = re.sub(r'[?%&]','', col)
        ret = ret + clean_col + ','               # oxjy167254jk,11-11-
2020,8:11:21,854a854,chow mein:,65,cash,sadabahar,delivered,5,awesome
experience,
    ret = ret[:-1]                                # oxjy167254jk,11-11-
2020,8:11:21,854A854,chow mein:,65,cash,sadabahar,delivered,5,awesome
experience
    return ret

def print_row(row):
    print row


cleaned_data = (
    p
    #| beam.io.ReadFromText(inputs_pattern, skip_header_lines=1)
    | beam.io.ReadFromPubSub(topic=inputs_pattern)
    | beam.Map(remove_last_colon)
    | beam.Map(lambda row: row.lower())
    | beam.Map(remove_special_characters)
    | beam.Map(lambda row: row+',1')          # oxjy167254jk,11-11-
2020,8:11:21,854a854,chow mein,65,cash,sadabahar,delivered,5,awesome
experience,1
```

```
)


delivered_orders = (
      cleaned_data
      | 'delivered filter' >> beam.Filter(lambda row:
row.split(',')[8].lower() == 'delivered')

)

other_orders = (
    cleaned_data
    | 'Undelivered Filter' >> beam.Filter(lambda row:
row.split(',')[8].lower() != 'delivered')
)

'''(cleaned_data
 | 'count total' >> beam.combiners.Count.Globally()             # 920
 | 'total map' >> beam.Map(lambda x: 'Total Count:' +str(x))   # Total
Count: 920
 | 'print total' >> beam.Map(print_row)

)

(delivered_orders
 | 'count delivered' >> beam.combiners.Count.Globally()
 | 'delivered map' >> beam.Map(lambda x: 'Delivered count:'+str(x))
 | 'print delivered count' >> beam.Map(print_row)
 )


(other_orders
 | 'count others' >> beam.combiners.Count.Globally()
 | 'other map' >> beam.Map(lambda x: 'Others count:'+str(x))
 | 'print undelivered' >> beam.Map(print_row)
 )'''

# BigQuery
client = bigquery.Client()

dataset_id = "{}.dataset_food_orders".format(client.project)

try:
      client.get_dataset(dataset_id)

except:
      dataset = bigquery.Dataset(dataset_id)  #

      dataset.location = "US"
      dataset.description = "dataset for food orders"
```

```
      dataset_ref = client.create_dataset(dataset, timeout=30)  # Make an
API request.

def to_json(csv_str):
    fields = csv_str.split(',')

    json_str = {"customer_id":fields[0],
                "date": fields[1],
                "timestamp": fields[2],
                "order_id": fields[3],
                "items": fields[4],
                "amount": fields[5],
                "mode": fields[6],
                "restaurant": fields[7],
                "status": fields[8],
                "ratings": fields[9],
                "feedback": fields[10],
                "new_col": fields[11]
                }

    return json_str

table_schema =
'customer_id:STRING,date:STRING,timestamp:STRING,order_id:STRING,items:STR
ING,amount:STRING,mode:STRING,restaurant:STRING,status:STRING,ratings:STRI
NG,feedback:STRING,new_col:STRING'

(delivered_orders
      | 'delivered to json' >> beam.Map(to_json)
      | 'write delivered' >> beam.io.WriteToBigQuery(
      delivered_table_spec,
      schema=table_schema,
      create_disposition=beam.io.BigQueryDisposition.CREATE_IF_NEEDED,
      write_disposition=beam.io.BigQueryDisposition.WRITE_APPEND,
      additional_bq_parameters={'timePartitioning': {'type': 'DAY'}}
      )
)

(other_orders
      | 'others to json' >> beam.Map(to_json)
      | 'write other_orders' >> beam.io.WriteToBigQuery(
      other_table_spec,
      schema=table_schema,
      create_disposition=beam.io.BigQueryDisposition.CREATE_IF_NEEDED,
      write_disposition=beam.io.BigQueryDisposition.WRITE_APPEND,
      additional_bq_parameters={'timePartitioning': {'type': 'DAY'}}
      )
)
```

```
def create_view():
    print('Creating VIEW thread ...')
    view_name = "daily_food_orders"
    dataset_ref = client.dataset('dataset_food_orders')
    view_ref = dataset_ref.table(view_name)
    view_to_create = bigquery.Table(view_ref)

    view_to_create.view_query = 'select * from `bigquery-demo-
285417.dataset_food_orders.delivered_orders` where _PARTITIONDATE =
DATE(current_date())'
    view_to_create.view_use_legacy_sql = False

    try:
        client.create_table(view_to_create)
    except:
        print 'View already exists'

from threading import Timer
t = Timer(25.0, create_view)
t.start()

from apache_beam.runners.runner import PipelineState
ret = p.run()
ret.wait_until_finish()
if ret.state == PipelineState.DONE:
    print('Success!!!')
else:
    print('Error Running beam pipeline')
```

OUTPUT :



food_daily

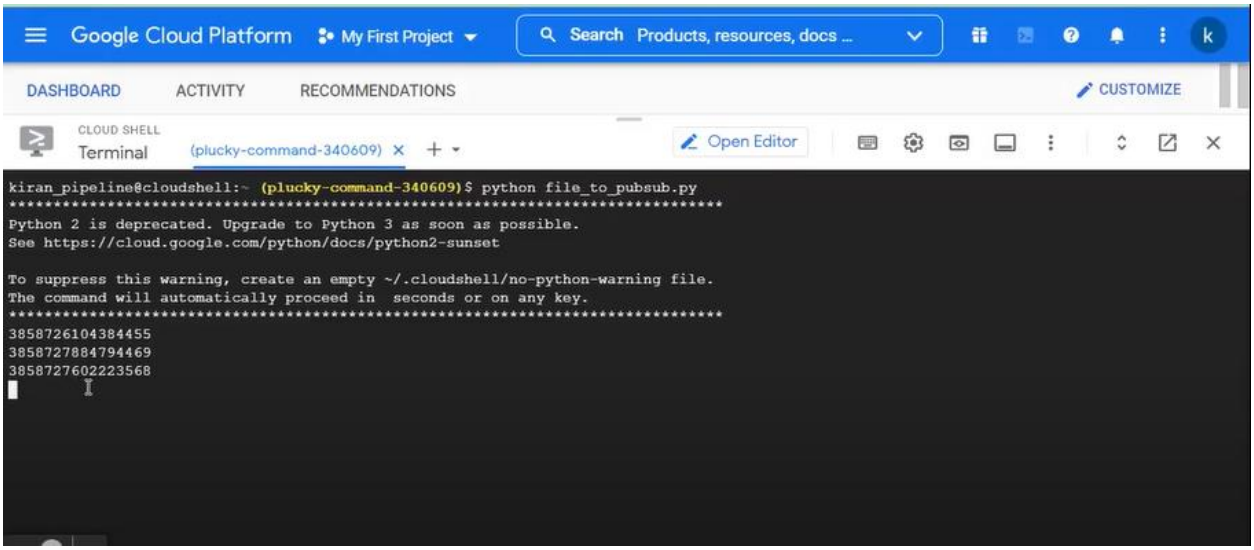
Saved to Drive

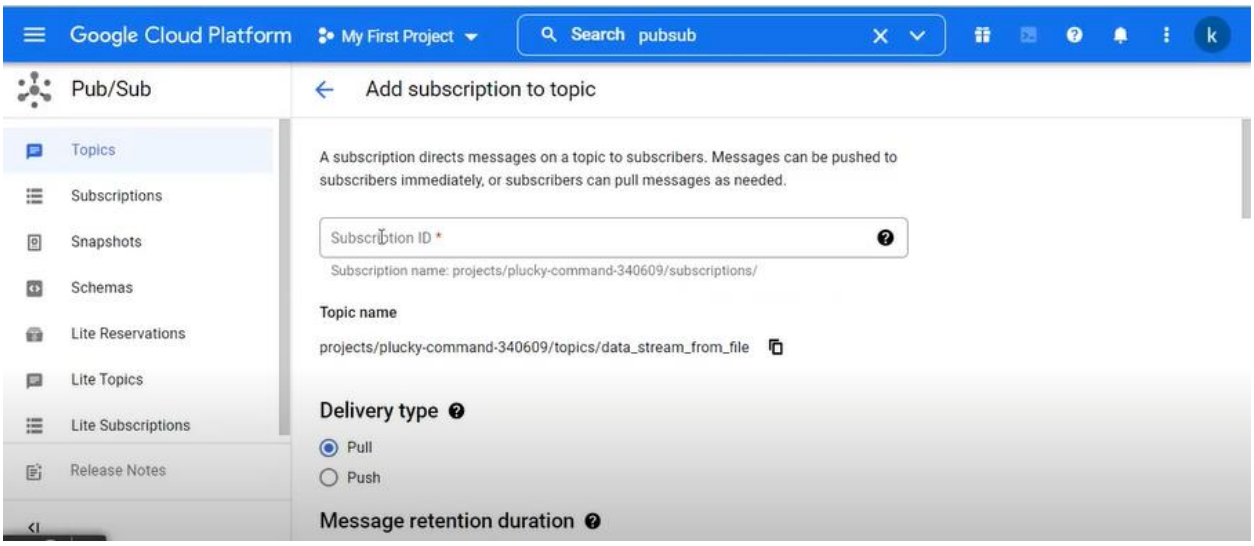
File Edit View Insert Format Data Tools Extensions Help Last edit was seconds ago

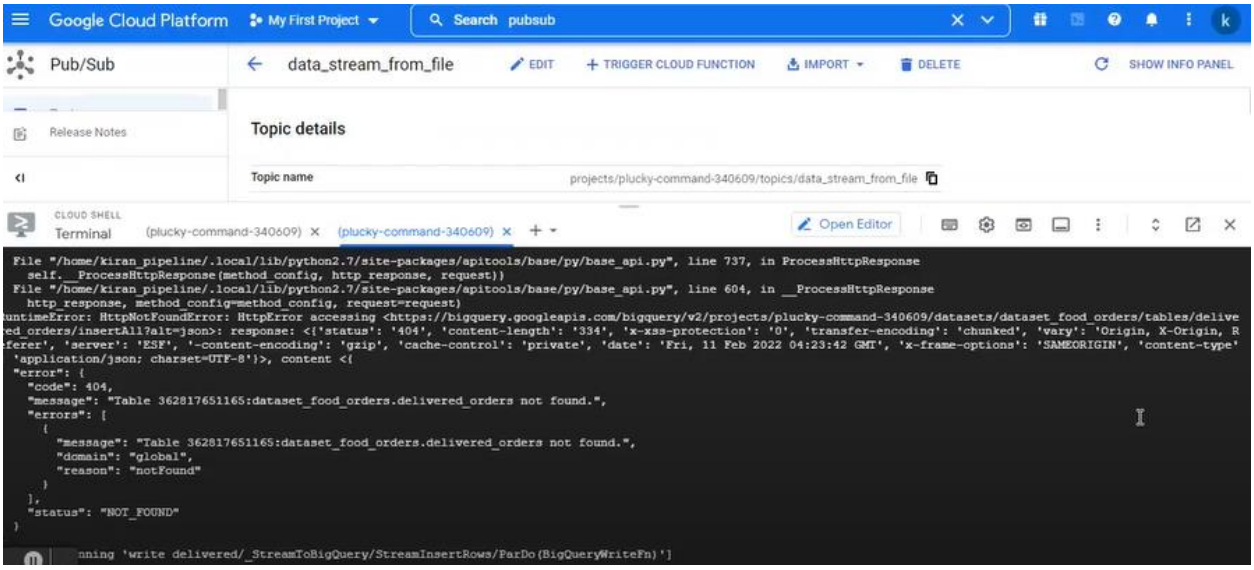
Share

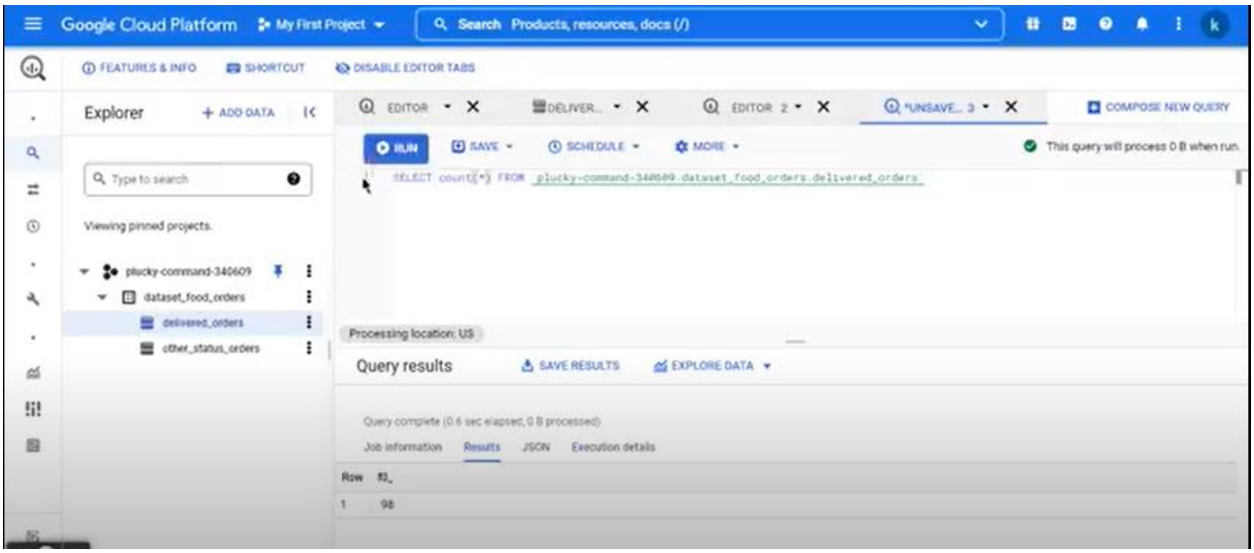
A1 Customer_id

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Customer_id	date	time	order_id	items	amount	mode	restaurnt	Status	ratings	feedback		
2	OXJY167254JK	11-11-2020	8.11.21	854A854	Chow M?ein:	65	Cash	Sadabahar	Delivered	5	Awesome experience		
3	JXJY167254JK	11-11-2020	8.12.20	654S654	PiZza:Manch?ur	197	Wallet	Emperial	Delivered	2	Late delivery		
4	JXJY167254JK	11-11-2020	8.12.21	2444454	Noodles:Pizza:R	97	Card	Sadabahar	Delivered	3	Stale food		
5	XVTR474839TP	11-11-2020	8.12.22	397T397	Fried Rice:salaD	46	Card	Emperial	Not Delivered	3	Complicated procedure		
6	UFDF355524DM	11-11-2020	8.12.23	428K428	noo%dles:	71	Card	Delhi pot	Delivered	1	Food not good		
7	FRBT691245BA	11-11-2020	8.12.24	437M437	Laccha Prantha:	29	Online	Eat out	delivered	1	Stale food		
8	QTSI285174SG	11-11-2020	8.12.25	106S106	Fried Rice:	59	Cash	Toit	Delivered	2	Complicated procedure		
9	SAMV824387M\	11-11-2020	8.12.26	751C751	Quiche:	28	Wallet	Khana Khazana	Delivered	1	Difficult to order		
10	JXJY167254JK	11-11-2020	8.12.27	5589654	Quiche:Crispy O	87	Card	Johnahs	Delivered	2	Late delivery		
11	JXJY167254JK	11-11-2020	8.12.28	8548654	Noodles:Pizza:R	97	Card	Onesta	Cancelled	1	Stale food		
12	JXJY167254JK	11-11-2020	8.12.29	954S654	Fryams:Manchu	77	Wallet	Johnahs	Delivered	3	Late delivery		
13	JXJY167254JK	11-11-2020	8.12.30	154S644	Quiche:Noodles:	123	Card	Eat out	On Hold	4	Will order again		
14	LRKI603654KL	11-11-2020	8.12.31	742G742	Roti:	86	Card	Little Cafe	Delivered	1	Difficult to order		
15	FRBT691245BA	11-11-2020	8.12.32	103Y103	Manchurian:	94	Online	Plan B	delivered	1	Late delivery		
16	FSAM929386AC	11-11-2020	8.12.33	208U208	Paneer tikka:	33	Wallet	Emperial	Delivered	1	Difficult to order		
17	FRBT691245BA	11-11-2020	8.12.34	819Q819	Regular:	86	Cash	Khana Khazana	Cancelled	4	Cheap and best		
18	UFDF355524DM	11-11-2020	8.12.35	107K107	Biryani:	50	Card	Eat out	Delivered	4	Delicious food		
19	FSAM929386AC	11-11-2020	8.12.36	671T671	Idli:	82	Card	Delhi pot	On Hold	5	Good service		
20	QTSI285174SG	11-11-2020	8.12.37	195K195	Crispy Onion Rir	40	Cash	Delhi pot	Delivered	1	High price		
21	QTSI285174SG	11-11-2020	8.12.38	952U952	pickle:	29	Online	Toit	Delivered	3	Complicated procedure		
22	LRKI603654KL	11-11-2020	8.12.39	833F833	Chikko shake:	33	Cash	Emperial	Delivered	3	Coupon applied		




Google Cloud Platform
My First Project
Search pubsub
Pub/Sub
Add subscription to topic
Topics
Subscriptions
Snapshots
Schemas
Lite Reservations
Lite Topics
Lite Subscriptions
Release Notes
A subscription directs messages on a topic to subscribers. Messages can be pushed to subscribers immediately, or subscribers can pull messages as needed.
Subscription ID *
Subscription name: projects/plucky-command-340609/subscriptions/
Topic name
projects/plucky-command-340609/topics/data_stream_from_file
Delivery type
Pull
Push
Message retention duration


Google Cloud Platform
My First Project
Search pubsub
Pub/Sub
data_stream_from_file
EDIT
TRIGGER CLOUD FUNCTION
IMPORT
DELETE
SHOW INFO PANEL
Release Notes
Topic details
Topic name
projects/plucky-command-340609/topics/data_stream_from_file
CLOUD SHELL
Terminal
(plucky-command-340609)
(plucky-command-340609)
Open Editor
File "/home/kiran_pipeline/.local/lib/python2.7/site-packages/apitools/base/py/base_api.py", line 737, in ProcessHttpResponse
self.__ProcessHttpResponse(method_config, http_response, request))
File "/home/kiran_pipeline/.local/lib/python2.7/site-packages/apitools/base/py/base_api.py", line 604, in __ProcessHttpResponse
http_response, method_config=method_config, request=request)
RuntimeError: HttpNotFoundError: HttpError accessing <https://bigquery.googleapis.com/bigquery/v2/projects/plucky-command-340609/datasets/dataset_food_orders/tables/delivered_orders/insertAll?alt=json>: response: <{'status': '404', 'content-length': '334', 'x-xss-protection': '0', 'transfer-encoding': 'chunked', 'vary': 'Origin, X-Origin, Referer', 'server': 'ESF', 'content-encoding': 'gzip', 'cache-control': 'private', 'date': 'Fri, 11 Feb 2022 04:23:42 GMT', 'x-frame-options': 'SAMEORIGIN', 'content-type': 'application/json; charset=UTF-8'}>, content <{
"error": {
"code": 404,
"message": "Table 362817651165:dataset_food_orders.delivered_orders not found.",
"errors": [
{
"message": "Table 362817651165:dataset_food_orders.delivered_orders not found.",
"domain": "global",
"reason": "notFound"
}
],
"status": "NOT_FOUND"
}
'write delivered/_StreamToBigQuery/StreamInsertRows/ParDo(BigQueryWriteFn)']


Google Cloud Platform
My First Project
Search Products, resources, docs (/)
FEATURES & INFO
SHORTCUT
DISABLE EDITOR TABS
Explorer
+ ADD DATA
EDITOR
DELIVER...
EDITOR 2
*UNSAVE... 3
COMPOSE NEW QUERY
RUN
SAVE
SCHEDULE
MORE
This query will process 0 B when run.
Type to search
Viewing pinned projects.
plucky-command-340609
dataset_food_orders
delivered_orders
other_status_orders
SELECT count(*) FROM plucky-command-340609.dataset_food_orders.delivered_orders
Processing location: US
Query results
SAVE RESULTS
EXPLORE DATA
Query complete (0.6 sec elapsed, 0 B processed)
Job information
Results
JSON
Execution details
Row
f0_
1
98